WEST

Help Logout Interrupt

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences | Cases

Search Results -

Terms	Documents
kmean$ or k adj1 (mean$ or median$ or prototyp$)	9

Database:
US Patents Full-Text Database
US Pre-Grant Publication Full-Text Database
JPO Abstracts Database
EPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L1

Refine Search

Recall Text Clear

Search History

DATE: Monday, September 23, 2002 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
	DB=TDBD; PLUR=YES; OP=OR		
L1	kmean$ or k adj1 (mean$ or median$ or prototyp$)	9	L1

END OF SEARCH HISTORY

WEST

Generate Collection | Print

Search Results - Record(s) 1 through 9 of 9 returned.

☐ 1. Document ID: NN9801207

L1: Entry 1 of 9 File: TDBD Jan 1, 1998

TDB-ACC-NO: NN9801207

DISCLOSURE TITLE: Automatic Categorization Tool for Organizing Brainstorming Comments

L1: Entry 1 of 9 File: TDBD Jan 1, 1998

DOCUMENT-IDENTIFIER: NN9801207
TITLE: Automatic Categorization Tool for Organizing Brainstorming Comments

Disclosure Text (2):
Technical Description: Team Kit/2 and Group Systems V are software packages used to support electronic meetings using the Team Focus technique. The technique often begins with a brainstorming process that generates many ideas in the form of typed comments. The next step in the technique is for the meeting attendees to group the comments into categories based on the ideas expressed in the comments. Typically, the group reviews the comments and places them into appropriate categories using a tool called the Idea Organizer (IO). This process can be very time consuming because the meeting is essentially a non-electronic meeting The Automated Categorization Tool is designed to be run as an intermediate step between idea generation and idea organization. It reads the files produced during an Electronic Brainstorming (EB) session and creates a set of files on the facilitator's workstation. Its output can be incorporated directly into the IO tool, in the same manner as the results of an EB session run with keywords is imported into an IO session. Text Pre-processing - In addition to the standard report file produced by the EB tool, the categorization tool uses three auxiliary files in its parsing process. Default versions of these files are provided with the software, but they can be customized with any standard editor (e.g., E, edlin, EPM, XEDIT, etc) to tailor them to specific requirements. Prefixes and suffixes. One file contains a list of prefixes and suffixes that are removed before comparing words. This supports a more intelligent comparison that is not sensitive to verb tense, plural forms, etc. An extensive set of suffixes is provided with the tool. Prefixes and suffixes are also removed from synonyms (see below) to allow greater flexibility when matching terms among comments. Synonyms. Another file contains a list of synonyms. Entries in this file create the semantic links between words that express the same or similar concepts but are not related orthographically. Also, this file can be used to store conjugations of irregular verbs that are not handled by simple suffix removal (e.g., take, took), common abbreviations (e.g., mgmt for management), and common misspellings. These constructs all serve to improve the correlation between the syntactic (the level of the cluster analysis) and semantic (the desired relationships) domains. Stop-words. The third auxiliary file used by this tool contains a list of common words (stop-words) that should not be used to judge comment similarity. These words tend to be poor indicators or similarity because they either occur too frequently, because they carry little or no meaning, or because their meaning is so context-dependent that different instances are likely to have different meaning (e.g., very, like). Parsing

- After the auxiliary files are read, the BE report file containing the participant's comments is parsed. The first pass through the report file selects the content words that will be used to characterize comments. Those words that occur less than three times are excluded from the analysis. The second pass builds a list of comments and their associated content words. The comments that are found not to contain any content words are placed in an "Uncategorized" category and are not included in the subsequent analysis. Clustering algorithm. A clustering algorithm based on the KMEANS algorithm (Spath, 1980) is used to partition the comments or clusters. The algorithm assigns each comment to a random cluster and then moves comments between the clusters to minimize the mean error of each cluster. The iterations stop when no additional moves decrease the cluster means.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc

☐ 2. Document ID: NA920691

L1: Entry 2 of 9 File: TDBD Jun 1, 1992

TDB-ACC-NO: NA920691

DISCLOSURE TITLE: Lossy Coding of Multi-band Continuous Tone Image Data.

L1: Entry 2 of 9 File: TDBD Jun 1, 1992

DOCUMENT-IDENTIFIER: NA920691
TITLE: Lossy Coding of Multi-band Continuous Tone Image Data.

Disclosure Text (1):
- Disclosed is a lossy coding method for data compression of multi-band continuous tone image data, such as data obtained from remote sensing instruments as HIRIS, AVIRIS and other radar imaging tools, which accounts for the correlation among the various images. - Images are sectioned into blocks (say, 128 by 128, 64 by 64, or any other size - not necessarily square or of sizes which are powers of two). Corresponding blocks of the various different bands are handled simultaneously. Some number K of prototypical images for the block are chosen. The number K depends on the statistics of the images, and has to be determined in each individual case. - In a preferred embodiment with K = 1, the prototypical image is the average of the corresponding blocks with some of the least significant bits dropped off. This prototypical image will be denoted by P1. For each image Ij, two constants aj,1 and bj,1 are computed. These are the approximate solutions (that is, a fixed bit accuracy of the actual solution) to the matrix-vector equation ***** SEE ORIGINAL FOR MATHEMATICAL EQUATIONS IN DOCUMENT ***** where Ij(k) and P1(k) are the respective pixel values of the images Ij and P1, and N is the total number of pixels in each image. Difference images are then formed ***** SEE ORIGINAL FOR MATHEMATICAL EQUATIONS IN DOCUMENT ***** In a preferred embodiment with K = 2, two prototypical images P1 and P2 are chosen. Three constants aj,2, bj,2, and cj,2 are computed, which are the approximate solutions (that is, a fixed bit accuracy of the actual solution) to the matrix-vector equation ***** SEE ORIGINAL FOR MATHEMATICAL EQUATIONS IN DOCUMENT ***** Difference images are then formed, This procedure can be extended to larger values of K, getting difference images with smaller and smaller variances. The total depth of a desirable procedure depends on the nature of the images and the number of correlated images processed simultaneously. One wishes to store few prototypical images and scaling and translation coefficients. In a preferred embodiment with only two prototypical images, one would store compressed

versions of the two prototypical images P1 and P2 together with the scaling coefficients aj,2 and bj,2 and the translation coefficients cj,2, plus all the difference images Dj,2 .

☐ 3. Document ID: NA9011433

L1: Entry 3 of 9 File: TDBD Nov 1, 1990

TDB-ACC-NO: NA9011433

DISCLOSURE TITLE: Quick VLSI CMOS Power Estimator.

L1: Entry 3 of 9 File: TDBD Nov 1, 1990

DOCUMENT-IDENTIFIER: NA9011433
TITLE: Quick VLSI CMOS Power Estimator.

Disclosure Text (1):
- Power estimation today tends to be based upon measurement in a simulated chip environment, or a rough estimation based upon number of circuits. - The advantage of this algorithm is that chip power and thus system-cooling requirements of small computers could be estimated early in the product cycle. - This algorithm estimates chip power from a VLSI chip schematic consisting of macros interconnected by buses. Each macro is assigned an average switching energy per output net (Es) and a probability of a macro output net switching/input transition (Ps). Ps might be a single number or a distribution correlated with other distributions, such as Ps for other macros. - USE IN TOP-DOWN SYSTEM DESIGN This algorithm is oriented toward use in an interactive top-down design system. The algorithm takes advantage of statistical tuning, by adjusting chip power to account for logic structure, logic levels, number of buses, bus widths, macro widths, multiple clocks, unclocked latches, different switching characteristics of different macros and statistical variations in chip fabrication and in chip operation. In top-down design the system is designed in a hierarchical fashion, starting with the highest level schematic, then detailing the schematic. This is followed by board and cable physical design, frames and covers, and then chip physical design. - The recommended time to use this algorithm is during board physical design, after the chip placement, as the driver switching energy is dependent upon driver output capacitance. Manhattan distance of off- chip nets could be used to estimate driver capacitance. - ALGORITHM (FORTRAN nomenclature will be used where appropriate to define the equations, and k denotes bit number k, where k = 1,2, ...,n.) 1. Ps(k,j)i) = mean probability of a macro output net (within output bus k) switching if there is an input transition to macro j or if there is a latch enable pulse. 2. Es(k,j) = When a macro output net (within output bus k) switches, Es is the average switching energy per output net. 3. Nb(k) = number of input buses for macro k 4. N(j) = the number of nets in bus j 5. Nl(k) = the mean number of logic levels for macro k 6. L(k) = the logic level at the output of macro k. 7. F(k) = The switching frequency of latch-oriented macro k, which is the clock frequency for clocked logic or an engineering estimation for unlatched logic. 8. fmax = the maximum switching frequency in the chip. All switching activity / cycle estimation is based upon fmax. 9. Tin(j,k) = mean number of switching transitions per fmax cycle of a net in the bus j input to macro k. 10. Tout (j,k) = mean number of switching transitions per fmax cycle of a net in the bus j output from macro k. 11. W(k) = average power generated by macro k. - The simplest form of

the algorithm is to: 1. Start with a schematic in the form of macros interconnected by buses. - a. Individual nets could be treated as buses of width 1 bit. - b. Individual logic blocks analog system blocks or latches could be treated as macros containing only one block. 2. Partition the chip macros into 2 types, latch-oriented and non- latch-oriented. - a. Latch-oriented macros are the start of logic paths. A latch- oriented macro may be clocked or unlocked. If a macro is clocked, it is assumed that all latches in that macro are clocked at the same frequency. - b. Non-latch-oriented macros receive inputs from logic paths that originate from latch-oriented macros or originate from chip inputs. - c. Assign a switching frequency F(k) to each latch-oriented macro k. - d. Assign a probability of switching Ps(k,j) to each output bus j connected to macro k. Ps(k,j) is an engineering estimate based upon the macro function and logic. - e. Assign a switching energy Es(k,j) to each output bus j connected to chip k. Es(k,j) is dependent upon net capacitances, which use average Manhattan distances for on-chip nets and board Manhattan distances (based upon chip placement) for off-chip nets. - f. Assign a mean number of logic levels NL(k) to each macro k. - g. Next identify the fastest average cycle latch macro on the chip. This could either be a clocked or unclocked macro. - h. Define fmax as this highest switching frequency. - i. For each output bus j from a latch macro k: Tout(k,j) = Ps(j,k) * f(k) / fmax j. Computing Tin for buses without dots, tristate or bidirectional Tin(k,j) = Tout(l,j), where macro(l) precedes macro(k) k. Computing Tin for buses with dots, tristate or bidirectional Tin(k,j) = The maximum Tout(l,j) of all macros(l) preceding macro k and with outputs Tout(l,j). - l. For each output bus j from a non-latch-oriented macro k: Tout(k,j) = Ps(k,j) * L(k) * Sum m over all input buses {Tin(k,m)} m. Then for latch or non-latch macro k L(k) = NL(k) + L(j) where L(j) is the highest logic level of a macro preceding macro k. For a latch-oriented macro L(j) = 0. - W(k) = Sum j over all output buses {Es(k,j)*Tout*fmax* N(j)} n. The total chip average power = Sum k over all macros {W(k)} EXTENSION Ps(k,j) and Es(k,j) could be distributions, and a Monte Carlo case-by-case computational technique similar to ASTAP statistical analysis could be added to the algorithm.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC | Draw Desc

☐ 4. Document ID: NN86102345

L1: Entry 4 of 9 File: TDBD Oct 1, 1986

TDB-ACC-NO: NN86102345

DISCLOSURE TITLE: Fast Labelling by Precomputation of Inter-Prototype Distances

L1: Entry 4 of 9 File: TDBD Oct 1, 1986

DOCUMENT-IDENTIFIER: NN86102345
TITLE: Fast Labelling by Precomputation of Inter-Prototype Distances

Disclosure Text (1):
- Speech may be considered a Euclidean space in which a spectral vector represents a segment of input speech and in which prototype vectors represent respective classes of sound or speech according to predefined characteristics. The present invention relates to methodology wherein the prototype vector that is closest to the spectral vector is determined without the need for computing a respective distance between the spectral vector and each prototype vector. In accordance with the invention, candidate prototype vectors are eliminated from consideration by using the triangle

law and precomputation. Let x be a spectral vector and let y1y2 ...yk be k prototype vectors. During labelling it is required to find the y nearest to xi, i.e., an I such that: where d(x,y) denotes a distance. Let aij be the precomputed inter-prototype distances: Then, 2d(x,yi) < aij implies that I=j is impossible. Then, each time a new d(x,yi) is computed, the following subset of prototypes is eliminated from further consideration; ***** SEE ORIGINAL DOCUMENT ***** The behavior of this algorithm has been simulated using prototypes of various dimensions n and a uniform distribution in n-dimensional space. That is, ***** SEE ORIGINAL DOCUMENT ***** The results are shown in the table on the preceding page. An entry in the table is the average number of distances computed divided by the maximum, i.e., the number of prototypes n.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC
Draw Desc | Clip Img

☐ 5. Document ID: NB82123705

L1: Entry 5 of 9 File: TDBD Dec 1, 1982

TDB-ACC-NO: NB82123705

DISCLOSURE TITLE: Pipelinable Data Tripartitioner by Means of the Median. December 1982.

L1: Entry 5 of 9 File: TDBD Dec 1, 1982

DOCUMENT-IDENTIFIER: NB82123705
TITLE: Pipelinable Data Tripartitioner by Means of the Median. December 1982.

Disclosure Text (1):
5p. A method for the pipelined generation of running medians for binary numbers of fixed length are extended not only to generate the median, but also to partition the input data into three classes, namely the classes of numbers larger than, equal to, and smaller than the median. In the pipelined implementation, the mechanism accepts an input binary number at every cycle and generates a median together with bit maps pointing to the three classes within the sample domain. - The median is important as a means to partition sampled data into three classes, namely - the 'upper' class, U, with numbers greater than the median; - the 'middle' class, V, of numbers equal in value to the median; - and the 'lower' class, W, with members smaller than the median. Example: Consider the inputs 69266 The median has the value 6; and the position for the upper class is indicated by U=01000, the position for the median values are found via V=10011, the position for the lower class is indicated by W=00100. The multiple occurrence of the median value in the sample is by no means rare. Any tie-breaking procedure could be invoked, if needed, by using the V-vector as a guide. - Often one desires the running median and associated information, through a window of fixed size. For example in processing the sequence. j= 123456789 Data(j)= 219133986 using a window of size N=5, we have successively Window(1) 11111 Sample 21913 Median 2 U 00101 V 10000 W 01010 Window(2) 11111 Sample 19133 Median 33 U 01000 V 00011 W 10100 Window(3) 11111 Sample 91339 Median 33 U 10001 V 00110 W 01000 Window(4) 11111 Sample 13398 Median 33 U 00011 V 01100 W 10011 Window(5) 11111 Sample 33986 Median 6 U 00110 V 00001 W 11000. Next, algorithms to yield the medians, also position vectors V and U, are described. - Let K be the bit matrix K of N rows and L columns formed by N unsigned binary numbers each of L bits. K(k) designates the kth column (NOT row) of K, counting from left to right. - The relative magnitude of the numbers can be found by examining the columns one at a

time, from left to right. The list of candidates for the median is successively narrowed, such that after k columns are processed, all surviving median candidates have the same leading k bits. Let U(k)=the bit map of the known members of the upper class, V(k)=the bit map of the surviving candidates for being the median , W(k)=the bit map of the known members of the lower class, and X(k)=the number of quantities expected to be greater than the median. - Clearly, if there are N=2n+1 numbers within the processing window, then before any processing, we have U(O)=an N-bit vector of all 0's V(0)=an N-bit vector of all 1's X(0)=n. - As K(k)=the kth 'Column' of the bit matrix is introduced, the survivors are further partitioned into two sets: The superior set showing a 1-bit in K(k); bit map=V(k-1) AND K(k); the inferior set showing a 0-bit in K(k); bit map=V(k-1) AND K(k). - Clearly any member of the superior set is larger than one in the inferior set, and the median is either in one or the other set. We form A(k)=Sum of the 1-bits in V(k-1) AND K(k) =The number of items in the superior set Q(k)=X(k-1) - A(k) and observe that If Q(k) is less than 0, the median is contained in the superior set. Hence, the superior set should survive as candidates; the inferior set belongs to the lower class; the other quantities are unchanged: U(k)=U(k-1) V(k)=V(k-1) AND K(k). - W(k)=W(k-1) OR (V(k-1) AND NOT K(k)) X(k)=X(k-1). Else, the inferior set should survive as candidates; the superior set belongs to the upper class; the number of survivors greater than the median is reduced; the other quantities remain unchanged: U(k)=U(k-1) OR (V(k-1) AND K(k)) V(k)=V(k-1) AND NOT K(k) W(k)=W(k-1) X(k)=Q(k). Then proceed to the next column. After L passes, all columns have been swept, and nonzero positions in V(L) show where the median is to be found. - It turns out that if the subtraction is in twos-complement arithmetic, then s(k)=the sign bit of Q(k) is precisely M(k), bit k of the median. - THE COMPLETE ALGORITHM FOR MEDIAN FINDING AND TRIPARTITION Word length=L Number of entries=N=2n+1 K(k)=kth column of data bits. - INITIALIZATION: X(0)=n V(0)=bit vector in length N, consisting of all 1's U(0)=bit vector in length N, consisting of all 0's W(0)=bit vector of length N, consisting of all 0's ALGORITHM DO 300 k=1 through L Step 1: Bring in K(k) A(k)=SUM OVER ((Vk-1) AND K(k)) Q(k)=X(k-1) - A(k) s(k)=1 if Q(k) is less than 0. Step 2: IF Q(k) LESS THAN 0, THEN U(k)=U(k-1) V(k) =V(k-1) AND K(k) W(k)=W(k-1) OR (V(k-1) AND NOT K(k)) X(k)=X(k-1) ELSE V(k)=V(k-1) AND NOT K(k) U(k)=U(k-1) OR (V(k-1) AND K(k)) V(k)=V(k-1) AND NOT K(k)) W(k)=W(k-1) X(k)=Q(k) 300 continue END ALGORITHM. - RESULTS Median=s(1), s(2), s(3), s(4)...s(L) Upper class bit map=U(L) Median bit map=V(L) Lower class bit map=W(L)=NOT (U(L) OR V(L)). - In the Algorithm above, we have used a control bit t(k)=1 for Step 1 t(k)=s(k) for Step 2.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC
Draw Desc

☐ 6. Document ID: NN81034740

L1: Entry 6 of 9 File: TDBD Mar 1, 1981

TDB-ACC-NO: NN81034740

DISCLOSURE TITLE: Amorphous CR(80)B(20) and CR(60)B(40) Extremely High Corrosion Resistant Alloys. March 1981.

L1: Entry 6 of 9 File: TDBD Mar 1, 1981

DOCUMENT-IDENTIFIER: NN81034740
TITLE: Amorphous CR(80)B(20) and CR(60)B(40) Extremely High Corrosion Resistant Alloys. March 1981.

Disclosure Text (1):
3p. In general, some improvement in the corrosion resistance of a metal alloy system results if it can be made amorphous. It is not considered to be a major effect. We have found that the corrosion resistance of pure chrome can be improved by a factor of 35,000 (see table below) by the addition of small amounts of boron to make it amorphous. This improvement persists down to at least 40 atomic percent boron. If the Cr-B alloys are annealed into a microcrystalline state, we see similar high corrosion resistance. When annealed to the microcrystalline state, the Cr(80)B(20) (amorphous) becomes chrome with boron interstitials. This makes the corrosion resistance even more remarkable considering that we are merely going from pure crystalline chrome towards a slightly expanded chrome lattice with boron interstitials. Composition Structure Corrosion Rate* Cr(80) B(20) Amorphous 0 Cr(80) B(20) Microcrystalline 210 Cr(80) B(20) Crystalline 4x10/4/ Cr (bulk) Large Crystals 7x10/6/ *Corrosion rate in A/day for samples immersed 5 days in concentrated HCL in order of increasing grain size. - We therefore conclude on the basis of this and some additional data (see table above) that by making the grain size extremely small or nonexistant (amorphous) a large improvement in the corrosion resistance results. The larger grains present edges, peaks, amorphous grain boundaries and different crystal facets which favor different oxide formation kinetics with an associated variation in thickness and density. This results in defects that are preferentially attacked. - It appears that the corrosion resistance of any metal that passivates (valve metals) spontaneously can be improved substantially by making the material microcrystalline or amorphous. This means that when it is desirable to maintain the characteristics, such as hardness, conductivity, etc., of a specific metal, one can make a major improvement in its corrosion resistance by adding a small, good glass-forming element, such as boron, and then annealing the sample only enough to cause the formation of a microcrystalline structure. For example amorphous Cr(80)B(20) annealed has a resistance and temperature coefficient of resistance like chrome with vastly improved corrosion resistance (see figure). - It also appears that the specific amorphous alloy system of Cr-B(20K), meaning that the amorphous phase is stable to very high temperatures (800 degrees K). At this temperature microcrystallites form which still show excellent corrosion resistance. (2) It does not corrode in an immersion test environment including 12N HCL, 3/1 HCL/H(2)O(2), 1/1 HNO(3)/HCL, 1/1 H(2)SO(4) and concentrated NaOH. (3) Since it is a simple binary, it can be fabricated in a variety of ways, such as sputtering, coevaporation, ion implantation, melt quenching and possibly electroplating. (4) It is easy to make amorphous. In sputtering, water-cooled substrates are adequate. This means that the actual substrate temperature is above room temperature. (5) No pitting occurs at anodic potentials (about one volt). Severe pitting was observed in the microcrystalline alloy. This has obvious applications wherever stray electric currents persist. (6) It is significantly harder than carbon tool steel. - To summarize, it has been found that in Cr-B and Al-B systems the corrosion resistance of any self-passivating metal can be substantially improved by decreasing its grain size. Also, the specific amorphous alloy of Cr-B (20

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC
Draw Desc | Clip Img

☐ 7. Document ID: NN7907855

L1: Entry 7 of 9 File: TDBD Jul 1, 1979

TDB-ACC-NO: NN7907855

DISCLOSURE TITLE: Direct Evaluation Of Queue Statistics In Closed Queueing Networks. July 1979.

L1: Entry 7 of 9 File: TDBD Jul 1, 1979

DOCUMENT-IDENTIFIER: NN7907855
TITLE: Direct Evaluation Of Queue Statistics In Closed Queueing Networks. July 1979.

Disclosure Text (1):
3p. An algorithm is described for computing queue statistics numerically for purposes of performance evaluation and system design. The algorithm covers the simple closed chain shown in the figure. Here, there is employed the known result which relates mean queue size n, mean waiting time w, and throughput lambda, as follows: n=lambda w. - The figure shows the closed queueing network. K is the population size, n(e)(K) is the mean queue size of server e; w(e)(K) is the mean queue size of server e; w(K) is the mean cycle time; and lambda (K) is the throughput. Equation (1) for the whole chain yields (see original) One more equation relating n(e)(K) and w(e)(K) is obtained by observing that a customer that just arrived at queue e finds n (K-1) customers ahead of him. In other words, a just arrived customer "sees" the equilibrium of the same system with one less customer. Assuming FIFO scheduling, there is obtained (see original) The above represents the algorithm to evaluate the network shown in the figure for a population size K=K(max). - The subject algorithm can easily be generalized to the case of multiple closed chains. In this case, equation (5) becomes (see original) where there is assumed two chains with population sizes K(1) and K(2) and where n(r,e,) is the average number of chain r customers at server e. Note that equation (6) becomes a good approximation to FCFS servers with different service demands by replacing tau(e) with tau(r,e,), the mean service time of chain r at server (see original). Also, with a suitable heuristic which relates n(K) with n(K-1), the algorithm can be made independent of K. This is an important advantage for the solution of networks with many closed chains which could not be solved so far.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC
Draw Desc | Clip Img

☐ 8. Document ID: NN78023806

L1: Entry 8 of 9 File: TDBD Feb 1, 1978

TDB-ACC-NO: NN78023806

DISCLOSURE TITLE: Word Recognition System Using Fast Dynamic Programming Alignment and Accurate Recognition Components. February 1978.

L1: Entry 8 of 9 File: TDBD Feb 1, 1978

DOCUMENT-IDENTIFIER: NN78023806
TITLE: Word Recognition System Using Fast Dynamic Programming Alignment and Accurate Recognition Components. February 1978.

Disclosure Text (1):
3p. A system is described for the optimization of temporal alignment and classification of speech utterances by application of separate metrics (i.e., units of measurement) for the temporal alignment and the classification components, thereby minimizing the number of operations necessary to "recognize" speech utterances. The system applies a temporal alignment based on part of the stored prototypes, and performs a classification based on the complete stored prototypes. -

Fig. 1 shows a two-stage word recognizer including an alignment stage 1, a recognition stage 2 and a store 3 for holding the parameters and prototypes used for such stages. - The alignment stage 1 employs an easily implementable metric: See original page 3806. where x(j)=j/th/ parameter, mu(ij)=j/th/ mean for i/th/ prototype, and n=number of parameters. Suitable parameters for this metric were obtained from a coarse spectral representation in terms of energy (dB) in the following frequency ranges 100-450 Hz) first formant region 450-750 Hz) 750-1500 Hz) second formant region 1500-2500 Hz) 2500-4500 Hz) fricative region together with overall spectral power (dB). - This metric g(i)(x(1)x(2),...,x(n)) is used with a dynamic programming algorithm to compute an optimal path 4 shown in Fig. 2. The data for each word consists of a time sequence of subpatterns (6 parameter values). During a training procedure, mu(ijk) is computed and stored in store 3 for each prototype i, parameter j, and time point k. The distance metric between time point k of prototype i and time point l of the candidate is See original page 3807 For each prototype, the corresponding alignment path 4 is passed on to the recognition stage 2 via line 5. - The recognition stage 2 uses a more complex metric, described in the preceding article. This metric follows the path obtained in the alignment stage. Because a given path is followed, processing time for this stage is much shorter than that for the alignment stage, even though a less complex metric is used there. An overall complex metric is obtained between the candidate time sequence and each of the proto prototype time sequences. The overall metric for prototype i is See original page 3808. and (y(lj)) is parameter data, m(ijk) are parameter means and delta(ijk) are the inverse of parameter standard deviations for prototype i, parameter j, time points k and l, and n=number of parameters. The recognition procedure identifies the candidate pattern as coming from class i, where H(i)

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC
Draw Desc | Clip Img

☐ 9. Document ID: NN78023804

L1: Entry 9 of 9 File: TDBD Feb 1, 1978

TDB-ACC-NO: NN78023804

DISCLOSURE TITLE: Fast, Accurate Metric in Time Warping, Dynamic Programming Recognizers. February 1978.

L1: Entry 9 of 9 File: TDBD Feb 1, 1978

DOCUMENT-IDENTIFIER: NN78023804
TITLE: Fast, Accurate Metric in Time Warping, Dynamic Programming Recognizers. February 1978.

Disclosure Text (1):
2p. A standard, currently popular solution to the matching of patterns requiring time warping is the use of dynamic programming, for example, for recognizing discrete utterances, as disclosed by F. Itakura 1]. - The optimum distance metric for normally distributed data with zero covariance terms is described by N. J. Nilsson 2] and is as follows: See original page 3804. - The proposed metric is an approximation of the above metric which retains both mean and variance terms: See original page 3805. - This metric has been found to be both fast and accurate for matching patterns requiring time warping in dynamic programming recognition procedures. The metric is accurate due to its retention of a measure of both the mean and standard deviation at each time point. A fast metric results from a reduction in the number of multiplications represented in the optimal metric. - The

data consists of a time sequence of subpatterns. During a training procedure, mu(ijk) and delta(ijk) are computed and stored for each prototype l, parameter j and time point k. - The distance metric between time point k of prototype i and time point l of the candidate is: See original page 3805. - As shown in the figure, the dynamic programming algorithm computes an optimal path and corresponding overall metric between the candidate time sequence and each of the prototype time sequences. The overall metric for prototype i is: See original page 3805. The recognition procedure identifies the candidate pattern as coming from class i, where H(i)

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC
Draw Desc | Clip Img

Generate Collection | Print

Terms	Documents
kmean$ or k adj1 (mean$ or median$ or prototyp$)	9

Display Format: TI,KWI | Change Format

Previous Page Next Page